How to Add or Remove Subform Instance at Runtime in Interactive Adobe Form

Applies to:

This article applies to WebDynpro Java/Adobe Interactive Forms development in SAP NetWeaver 7.0 and Adobe LiveCycle Designer 8.0 or higher release. Demo application created for this article has been developed in SAP NetWeaver 7.0 EHP1. For more information, visit the Web Dynpro Java homepage.

Summary

This article tells about how to add or remove subform instance at runtime in Interactive Adobe Form development.

Author: Suresh Thiyagarajan

Company: Tata Consultancy Services

Created on: 8 September 2010

Author Bio

Suresh Thiyagarajan is a SAP NetWeaver Consultant working for Tata Consultancy Services, India. His areas of expertise in SAP technologies include WebDynpro Java, WebDynpro ABAP, Enterprise Portal and Adobe Interactive Form development.

Table of Contents

Introduction

This article is the first of a three part series on Adobe Interactive Forms development. While searching on SDN for help documents on Adobe forms development for a past project, I was surprised at the absence of documentation for many features of Adobe forms in WebDynpro. This prompted me to record the learning that I had with Adobe forms and publish them as articles for the benefit of others like me. I hope some of these articles would be found helpful. The following topics are covered through this series.

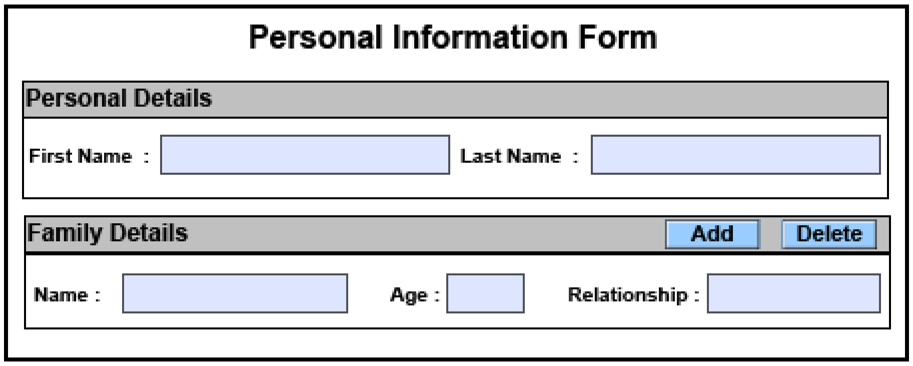
Part 1: How to add or remove subform instance at runtime in Interactive Adobe Form.

Part 2: How to extract data from an online Interactive Form using XML parsing in WebDynpro Java.

Part 3: How to extract data from an offline Interactive Form using XML parsing in WebDynpro Java.

Scenario

In this article, the below shown sample screen will be used to illustrate how to add/delete rows dynamically at runtime. This personal form allows a user to add as many rows as needed under the section for family details. Clicking on 'Add' button adds a row, while clicking on 'Delete' button deletes the last row.



Prerequisites

The following software needs to be installed before we can start developing Adobe Interactive Form,

- Adobe Livecycle Designer (*client side installation*)
- Adobe Acrobat Reader (*client side installation*)
- NetWeaver Developer Studio (*client side installation*)
- Zero Client Installation (*ZCI is only for 'Native' type forms*)

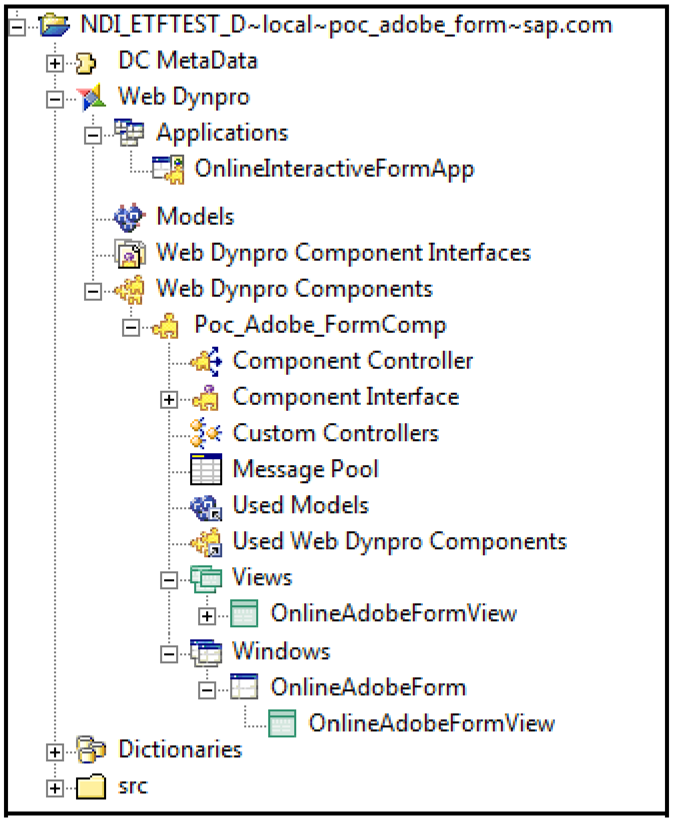
  ZCI should be configured separately on the server for all releases prior to NetWeaver 7.0 SPS10. For SAP NetWeaver 7.0 SPS10 or higher, ZCI comes as a default option.

  For more info, please visit the following link http://help.sap.com/saphelp_nw70/helpdata/en/2c/241a427ff6db2ce10000000a1550b0/frameset.htm
- Active Component Framework Plug-In (*ACF is only for 'ActiveX' type forms*)

  ACF plug-in is required to develop the Interactive Adobe Form using NetWeaver 7.0 SP09 or lower. From the release of SAP NetWeaver 7.0 SPS10 or higher, SAP recommends using *Native* type forms instead of *ActiveX* type.
- Adobe Document Service installed on Web Application Server (s*erver side installation*)
- Experience in WebDynpro Java development is required while familiarity with Adobe Interactive Forms would be an advantage.

Creating a WebDynpro Development Component

Let us start off by creating a WebDynpro DC as shown below. We shall be using the '*OnlineAdobeFormView*' view to design our Adobe form.

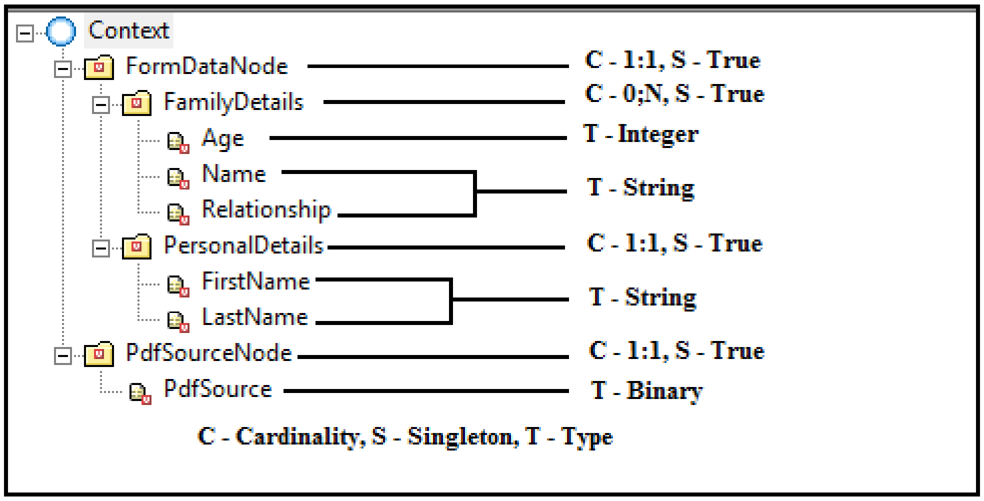


```
NDI_ETFTEST_D~local~poc_adobe_form~sap.com
  DC MetaData
  Web Dynpro
    Applications
      OnlineInteractiveFormApp
    Models
    Web Dynpro Component Interfaces
    Web Dynpro Components
      Poc_Adobe_FormComp
        Component Controller
        Component Interface
        Custom Controllers
        Message Pool
        Used Models
        Used Web Dynpro Components
        Views
          OnlineAdobeFormView
        Windows
          OnlineAdobeForm
            OnlineAdobeFormView
  Dictionaries
  src
```

Creating context structure

Before we can start with the design of the form, we need to create nodes and attributes to store the runtime data. In this particular example, nodes have been created in the component controller context and mapped to the view.

InteractiveForm UI generates XML data out of the WebDynpro context data (node and attributes bound to it) at runtime. In cases where the there is no dynamic programming involved, WebDynpro takes care of the parsing between XML and context data with the help of WebDynpro API(*WDInteractiveFormHelper*). In scenarios where the quantity of data holders is truly dynamic like adding or removing the subform instance (as in this article) or table row at runtime, XML parsing methods need to be used to extract the data from Interactive Form.



```
Context
  FormDataNode          C - 1:1, S - True
    FamilyDetails       C - 0;N, S - True
      Age               T - Integer
      Name
      Relationship      T - String
    PersonalDetails     C - 1:1, S - True
      FirstName
      LastName          T - String
  PdfSourceNode         C - 1:1, S - True
    PdfSource           T - Binary
```

C - Cardinality, S - Singleton, T - Type

FormDataNode – This is the root node bound to *dataSource* property of the InteractiveForm UI element. It is mandatory to bind the *dataSource* property to a node with cardinality 1:1. This node can have further child nodes which would then be accessible in the *Data View* of Adobe Livecycle Designer.

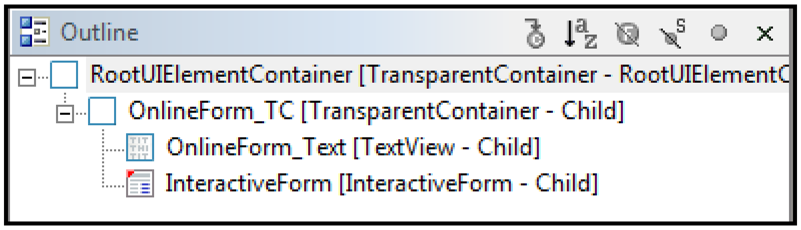
FamilyDetails – This node contains attributes *Name, Age* and *Relationship*. These attributes will be used for binding the corresponding input fields under the section of *Family Details.* As mentioned above*, w*hen adding or deleting the rows at runtime, node elements will not be created for this node automatically. Instead the XML Source will be updated with node structure and user entered data in the Adobe Form. XML parsing methods are then used to extract this information and add elements to the node.

PersonalDetails – This node contains attributes *FirstName* and *LastName*. These attributes are used to bind the input fields under section *Personal Details*. As the cardinality of this node is 1:1 the user entered data can be retrieved by directly by accessing the node element (WebDynpro takes care of the XML parsing with the help of WebDynpro API (*WDInteractiveFormHelper)*).

PdfSourceNode – This node contains the attribute *PdfSource* of type *Binary*. The *pdfSource* property of the InteractiveForm UI element is bound to this attribute and it holds the binary file of the PDF document at runtime.

Creating UI elements in the view

Open the *Layout* tab of the *OnlineAdobeFormView* and add an *InteractiveForm* UI element as shown below.

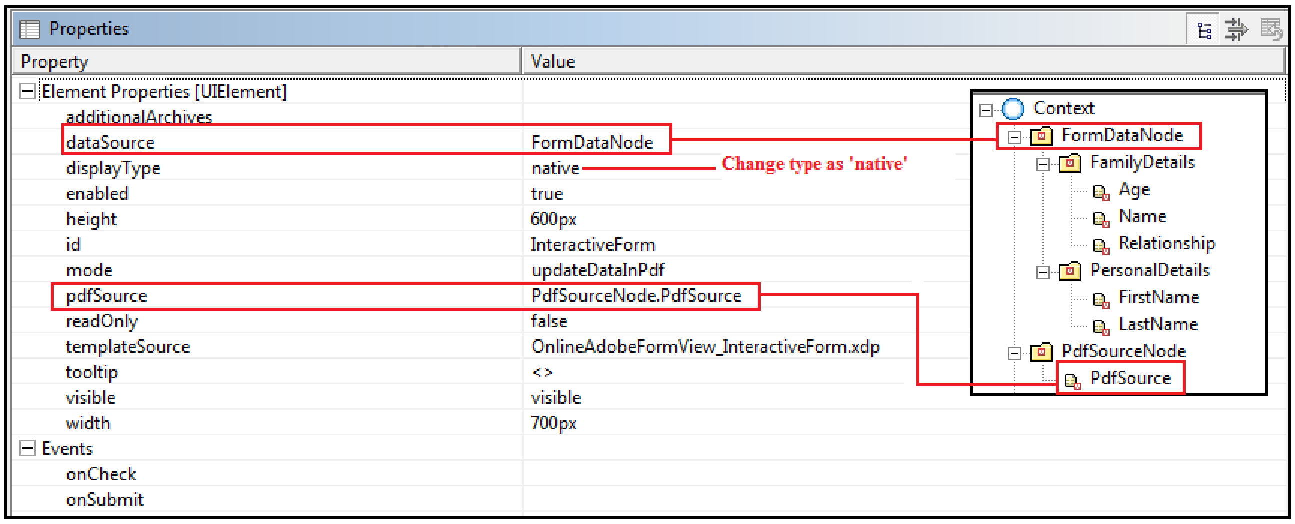


```
RootUIElementContainer [TransparentContainer - RootUIElementC
  OnlineForm_TC [TransparentContainer - Child]
    OnlineForm_Text [TextView - Child]
    InteractiveForm [InteractiveForm - Child]
```

Data binding between UI element and context

Select *InteractiveForm* UI from *Outline* tab and go to *Properties* perspective.

Change the *dataSource*, *displayType* and *pdfSource* as shown below.



Property	Value
Element Properties [UIElement]	
additionalArchives	
dataSource	FormDataNode
displayType	native
enabled	true
height	600px
id	InteractiveForm
mode	updateDataInPdf
pdfSource	PdfSourceNode.PdfSource
readOnly	false
templateSource	OnlineAdobeFormView_InteractiveForm.xdp
tooltip	<>
visible	visible
width	700px
Events	
onCheck	
onSubmit	

```
Context
  FormDataNode
    FamilyDetails
      Age
      Name
      Relationship
    PersonalDetails
      FirstName
      LastName
  PdfSourceNode
    PdfSource
```

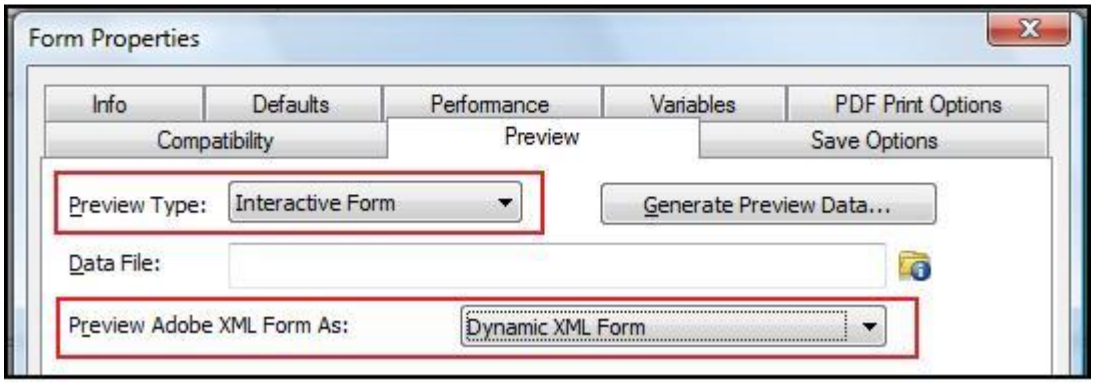
Designing the Interactive Adobe Form

Next step is to design the form itself. Before we start developing the form, we need change some default properties in Adobe Livecycle Designer to enable the creation of dynamic form.

Making a Dynamic Form

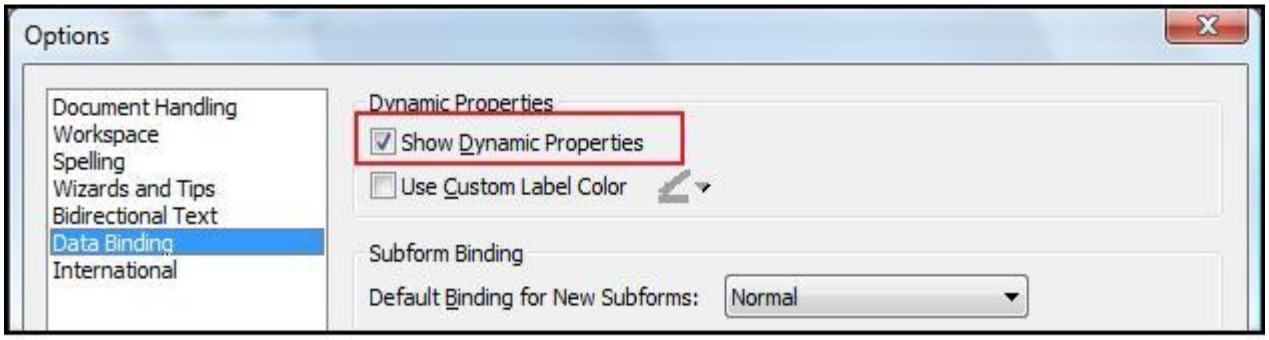
Select *InteractiveForm* UI element → *Right Click* → *Edit* to open the form in Adobe Livecycle Designer. Follow the below steps to enable the dynamic form properties.

1. In Adobe Livecycle Designer, *Go to* → *Edit* → *Form Properties* → *Preview (Tab)*. Change the preview type property as shown below.



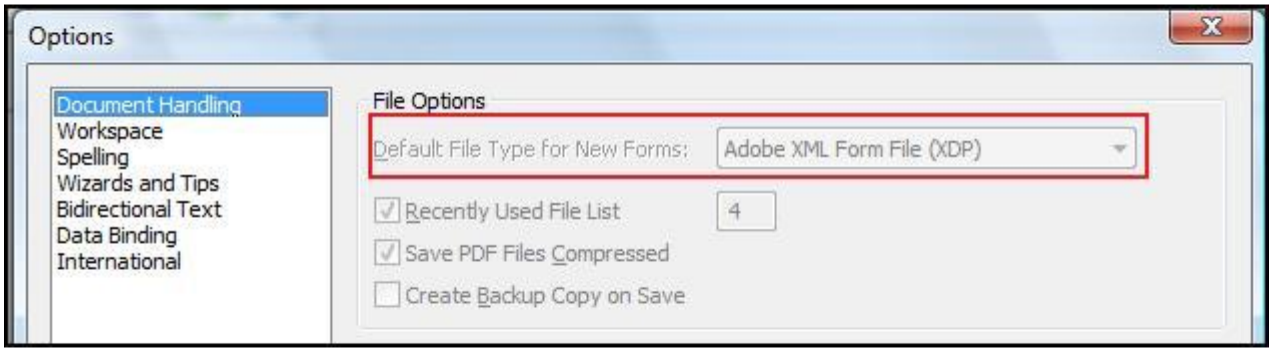
Preview Type: Interactive Form
Data File:
Preview Adobe XML Form As: Dynamic XML Form

2. In Adobe Livecycle Designer, *Go to* → *Tools* → *Options* → *Data Binding*. Check the checkbox for *Show Dynamic Properties*.



Dynamic Properties
☑ Show Dynamic Properties
☐ Use Custom Label Color
Subform Binding
Default Binding for New Subforms: Normal

3. In Adobe Livecycle Designer, *Go to* → *Tools* → *Options* → *Document Handling*. Make sure the type selected is *Adobe XML Form File* (XDP).



File Options
Default File Type for New Forms: Adobe XML Form File (XDP)
☑ Recently Used File List 4
☑ Save PDF Files Compressed
☐ Create Backup Copy on Save

4. In the view '*OnlineAdobeFormView'*, write the below piece of code in *wdModifyView ()* method. This code should be written in the view which belongs to the InteractiveForm UI has been added. Dynamic PDFs are capable of changing their layout/behavior at runtime based on the action/event triggered in the document. Should you need to deal with a dynamic PDF, you need to set this flag to *true* while creating the PDF document using the interface *IWDPDFDocumentInteractiveFormContext*.

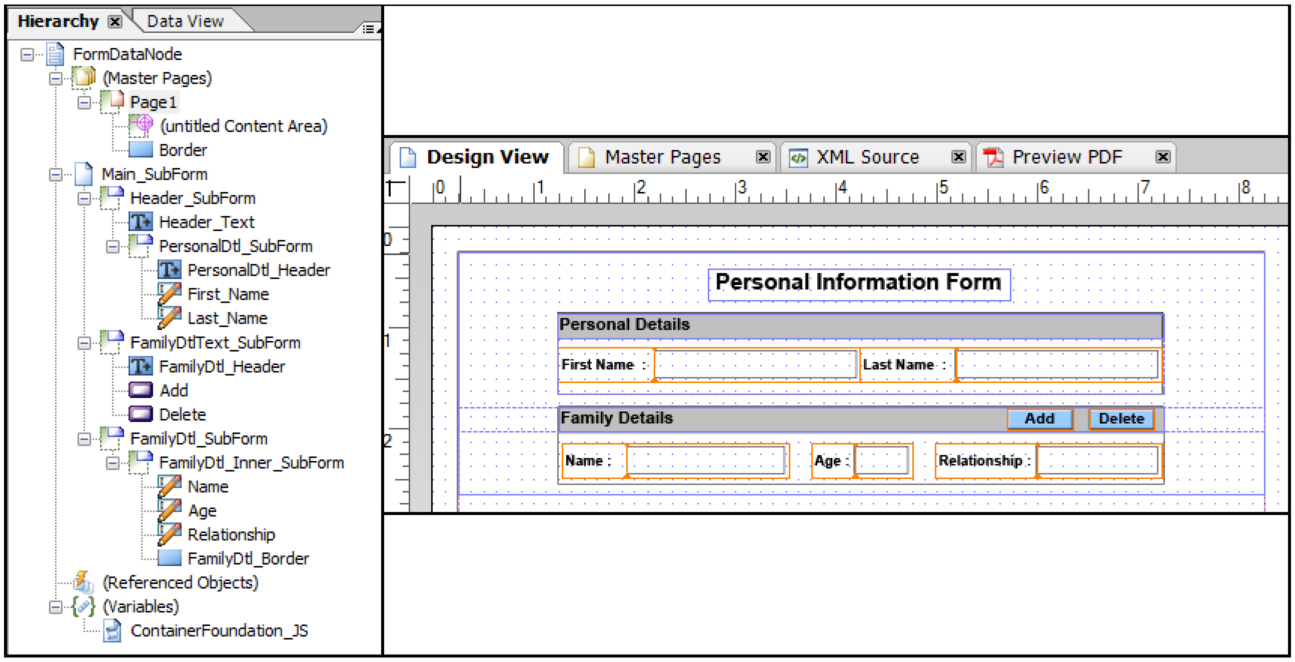
```
public static void wdDoModifyView(IPrivateOnlineAdobeFormView wdThis,
IPrivateOnlineAdobeFormView.IContextNode wdContext, com.sap.tc.webdynpro.progmodel.api.IWDView
view, boolean firstTime)
 {
   //@@begin wdDoModifyView
   if(firstTime)
   {
        IWDInteractiveForm interactiveForm = (IWDInteractiveForm)view.getElement("InteractiveForm");
        IWDController controller = interactiveForm.getView();
        IWDPDFDocumentInteractiveFormContext formContext = WDPDFDocumentFactory.
                        getDocumentHandler(controller,"InteractiveForm").getDocumentContext();
        formContext.setDynamic(true);
   }
   //@@end
```

Screen design

To open the Adobe Form designer, Go to *Outline* tab → select *InteractiveForm* UI → *Right Click* → *Edit*.

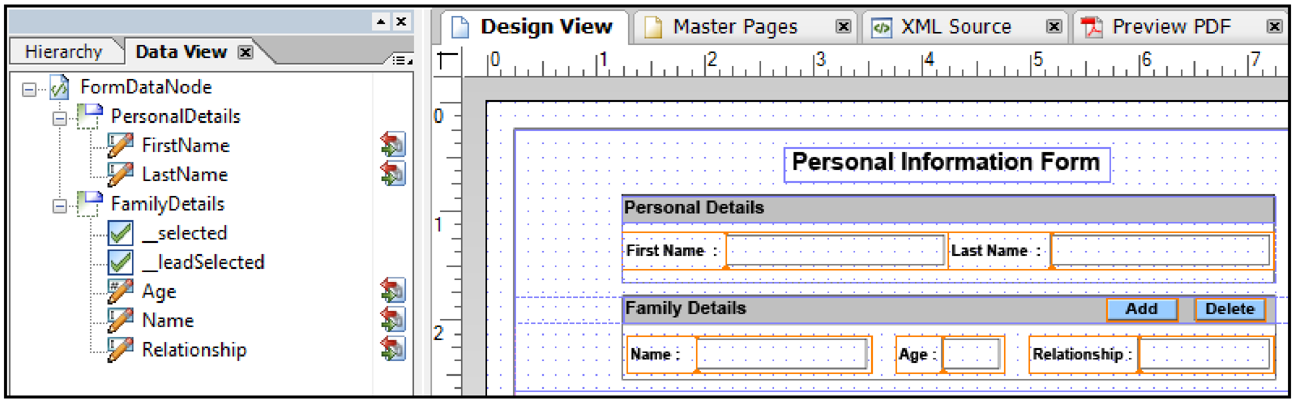
The Adobe Livecycle Designer opens in the SAP NetWeaver Developer Studio with the *Design View* as the default view in the center. Many UI elements are available to design the form, such as *text fields*, *buttons* and *checkboxes*. These could be added to the design from the *Library* tab or by using the drag and drop function. The *Data View* of the Adobe Designer provides the context node *FormDataNode* to which the *dataSource* property of the InteractiveForm UI element is bound.

Upon completion of design, the screen should look like as shown below. The left side of the screen shows the hierarchy of UI elements and is helpful in designing the form.



Binding the form elements to WebDynpro context

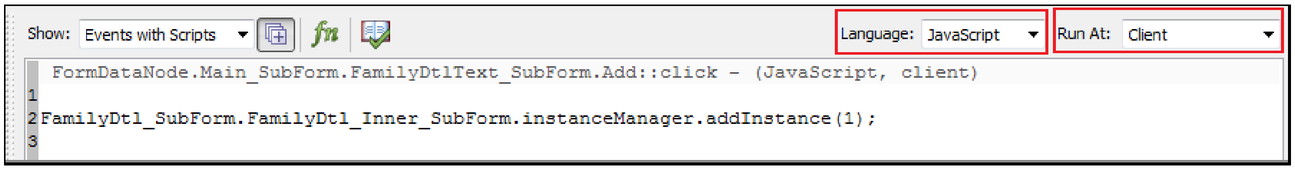
The *Data View* of the Adobe Designer shows the context node *FormDataNode* which is bound to the *dataSource* property of the InteractiveForm UI element. Once the screen design has been completed, the next step is to map the data between *Design View* and *Data View*. Go to *Data View* palette, drag and drop value attribute into corresponding input field in the *Design View*. After mapping the context attributes the screen will look as shown below.



Java Script for adding and removing subform instance

The next step is to write java script methods for adding and removing the subform instance and invoke them upon clicking 'Add' and 'Delete' buttons respectively. Before writing the script, couple of settings needs to be changed in the script editor - the option for Language has to be set as '*JavaScript'* and the *Run at* property has to be set as '*Client'.* You can find the options *Language* and *Run at* in the script editor as shown below (marked in red color).

Select button 'Add' from *Design View*, go to script editor and select the *'Click'* event from dropdown. Inside the event, write the below script to add subform instance at runtime.

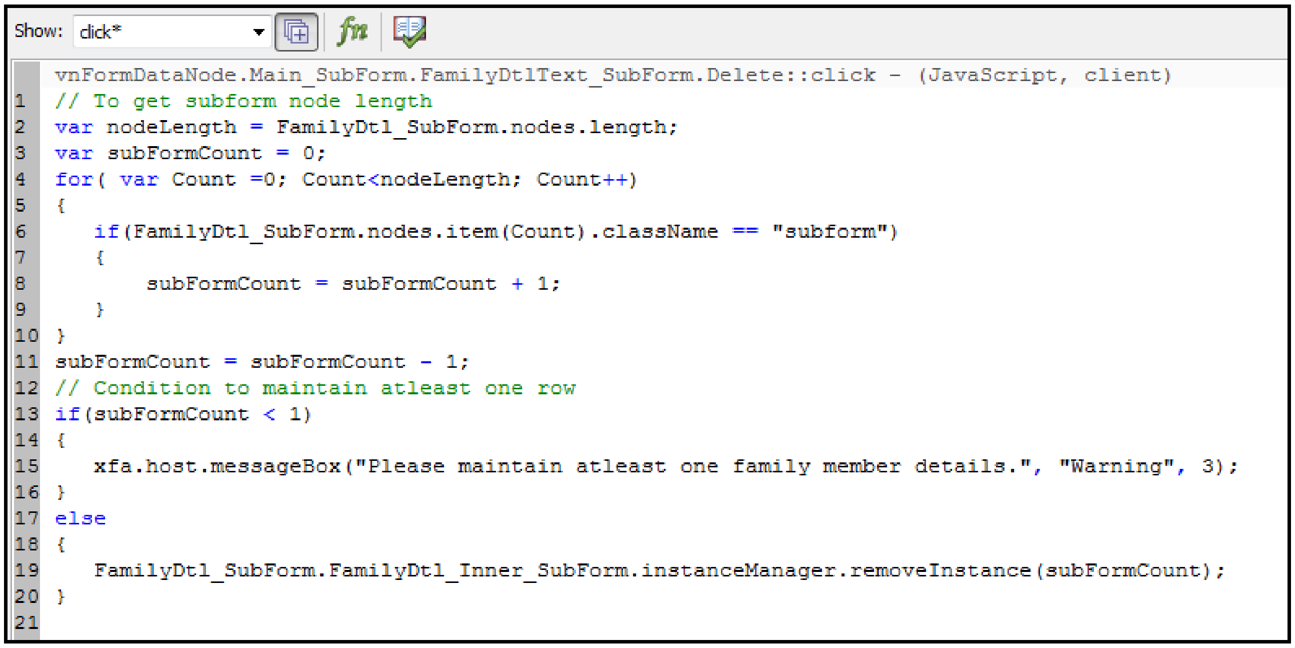

Show: Events with Scripts
Language: JavaScript
Run At: Client

```
 FormDataNode.Main_SubForm.FamilyDtlText_SubForm.Add::click - (JavaScript, client)

FamilyDtl_SubForm.FamilyDtl_Inner_SubForm.instanceManager.addInstance(1);

```

Code Snippet

```
FamilyDtl_SubForm.FamilyDtl_Inner_SubForm.instanceManager.addInstance(1);
```

Select button *'Delete'* from *Design View*, go to script editor and select the *'Click'* event from dropdown. Inside the event, write the below lines of code to remove subform instance at runtime.



```
vnFormDataNode.Main_SubForm.FamilyDtlText_SubForm.Delete::click - (JavaScript, client)
// To get subform node length
var nodeLength = FamilyDtl_SubForm.nodes.length;
var subFormCount = 0;
for( var Count =0; Count<nodeLength; Count++)
{
   if(FamilyDtl_SubForm.nodes.item(Count).className == "subform")
   {
       subFormCount = subFormCount + 1;
   }
}
subFormCount = subFormCount - 1;
// Condition to maintain atleast one row
if(subFormCount < 1)
{
   xfa.host.messageBox("Please maintain atleast one family member details.", "Warning", 3);
}
else
{
   FamilyDtl_SubForm.FamilyDtl_Inner_SubForm.instanceManager.removeInstance(subFormCount);
}
```

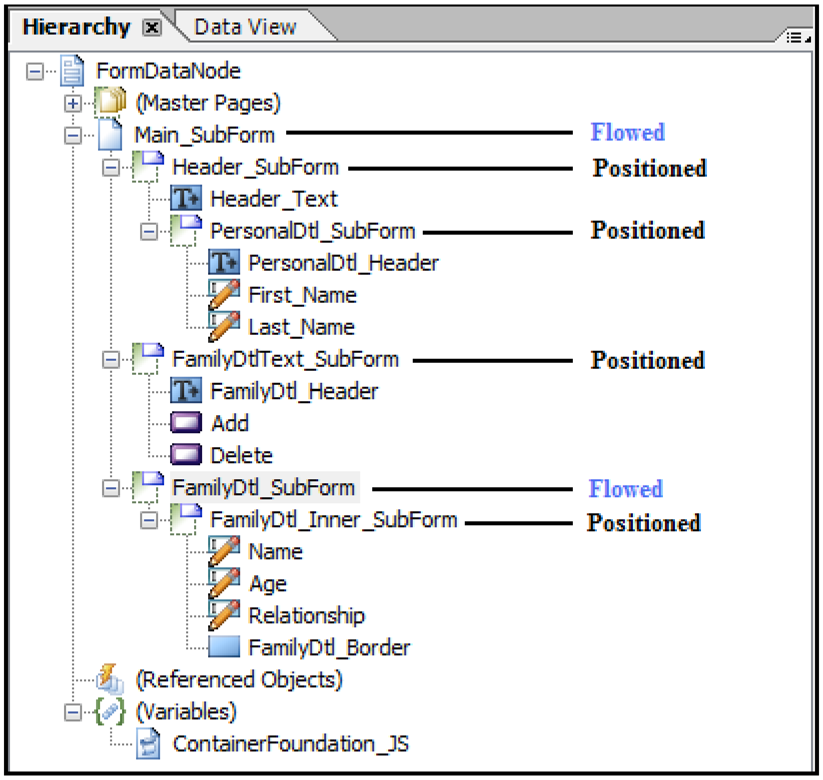
Code Snippet

```
// To get subform node length
 var nodeLength = FamilyDtl_SubForm.nodes.length;
 var subFormCount = 0;
 for( var Count =0; Count<nodeLength; Count++)
 {
        if(FamilyDtl_SubForm.nodes.item(Count).className == "subform")
        {
                subFormCount = subFormCount + 1;
        }
 }
 subFormCount = subFormCount - 1;
 // Condition to maintain atleast one row or else will display the warning message
 if(subFormCount < 1)
 {
        xfa.host.messageBox("Please maintain atleast one family member details.", "Warning", 3);
 }
 else
        FamilyDtl_SubForm.FamilyDtl_Inner_SubForm.instanceManager.removeInstance(subFormCount);
```

Make the subform as flowed

This personal form should allow a user to add as many rows as needed under the section for *Family Details*. Clicking on 'Add' button should add a row, while clicking on 'Delete' button should delete the row from last. To achieve this, we have to make the content type for both the subforms *Main_SubForm* and *FamilyDtl_SubForm* as *Flowed* and also enable the property *'Repeat Subform for Each Data Item'* for the subform *FamilyDtl_Inner_SubForm*. These changes have been explained in detail below.

By default the content type of each subform is set as *Positioned*. In our example, the content type of some subforms need to be set as *Flowed* in *Hierarchy* tab. The below diagram lists down the subforms whose content type needs to be updated.

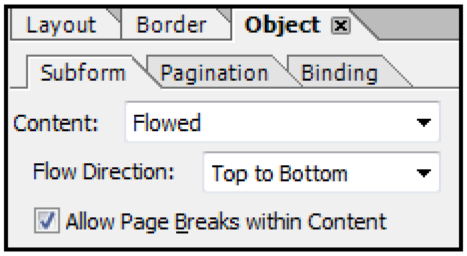


```
FormDataNode
  (Master Pages)
  Main_SubForm ———————— Flowed
    Header_SubForm ———————— Positioned
      Header_Text
      PersonalDtl_SubForm ———————— Positioned
        PersonalDtl_Header
        First_Name
        Last_Name
    FamilyDtlText_SubForm ———————— Positioned
      FamilyDtl_Header
      Add
      Delete
    FamilyDtl_SubForm ———————— Flowed
      FamilyDtl_Inner_SubForm ———————— Positioned
        Name
        Age
        Relationship
        FamilyDtl_Border
  (Referenced Objects)
  (Variables)
    ContainerFoundation_JS
```

Changing the subform content type

Step 1: First select the subform *FamilyDtl_SubForm* from *Hierarchy* palette → select *Object* palette → click *Subform* tab → change content type as *Flowed.*

Object palette will look as shown below.



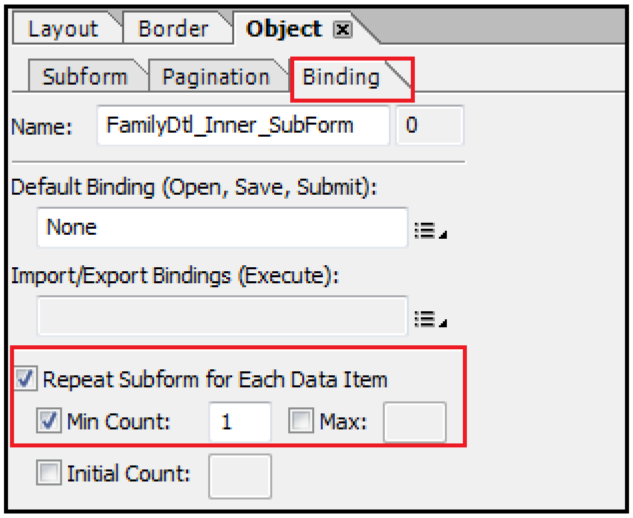
Content: Flowed

Flow Direction: Top to Bottom

[✓] Allow Page Breaks within Content

Step 2: Next step is to select the subform *FamilyDtl_Inner_SubForm* from *Hierarchy* palette → select *Object* palette → click *Binding* tab. Select the checkbox '*Repeat Subform for Each Data Item*'. It is mandatory to enable this option to flow the subform, while adding or removing the subform instance at runtime.

FamilyDtl_Inner_SubForm type should be *Positioned.*



Name: FamilyDtl_Inner_SubForm 0

Default Binding (Open, Save, Submit): None

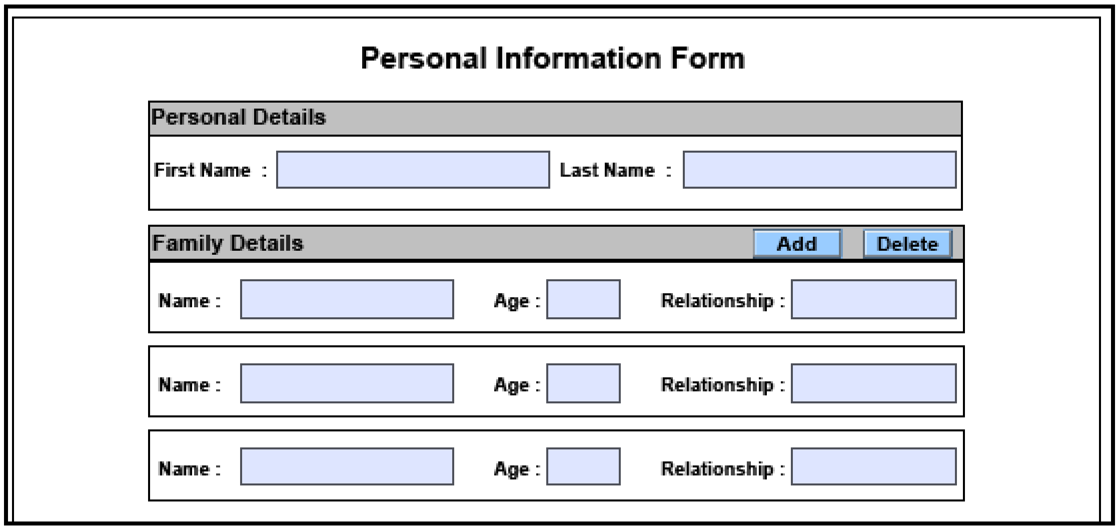
Import/Export Bindings (Execute):

☑ Repeat Subform for Each Data Item

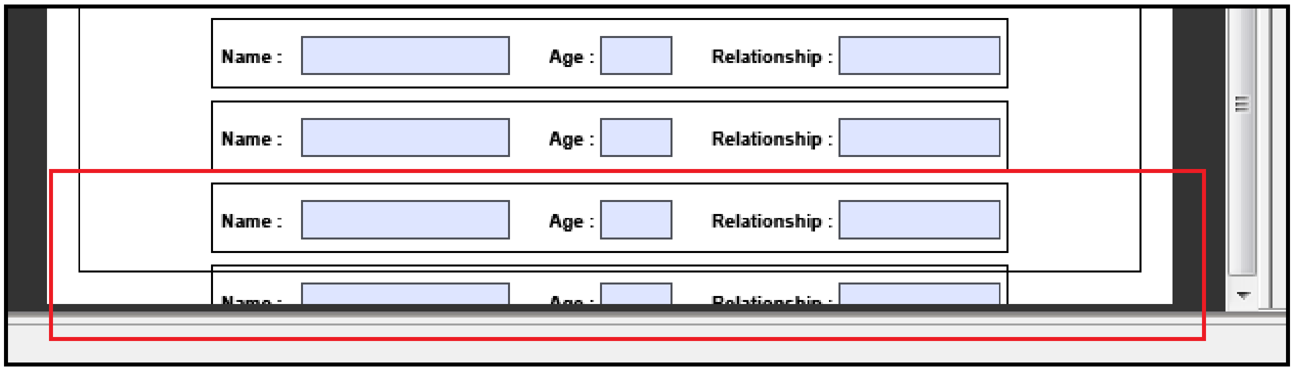
☑ Min Count: 1 ☐ Max:

☐ Initial Count:

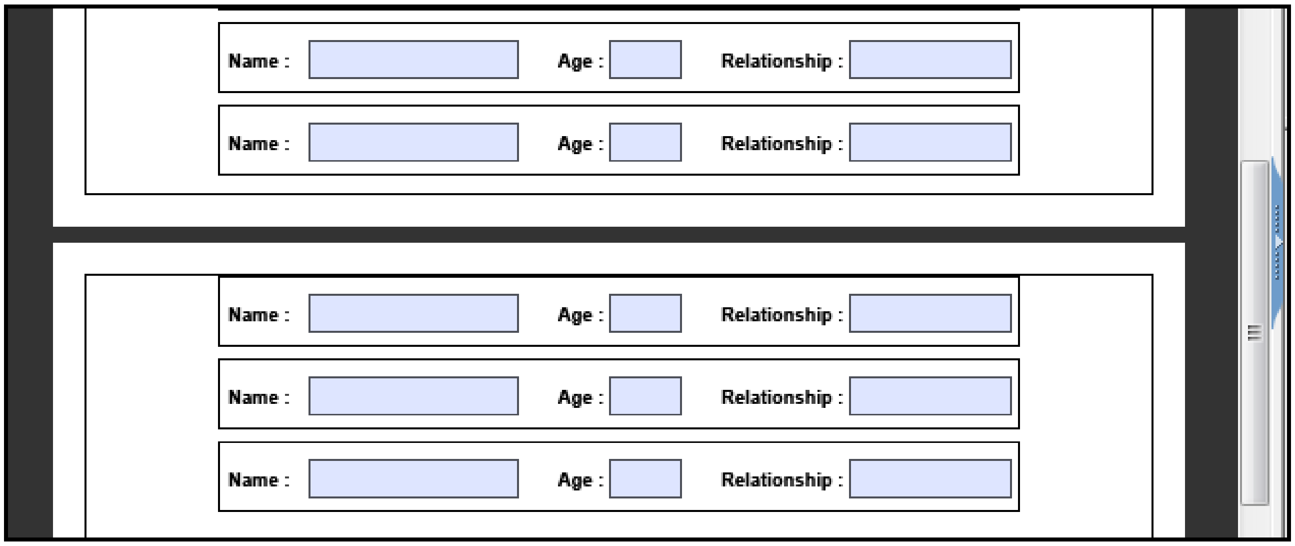
After completing Step 1 and Step 2, the add and delete functinalities can be tested in the *PDF Preview* view.



Note: When you try to add rows continuously in the first page and if the page exceeds the layout, the added row will not flow automatically into the second page. Instead it will be added outside the page layout as shown below. To avoid this issue, please do the step 3.

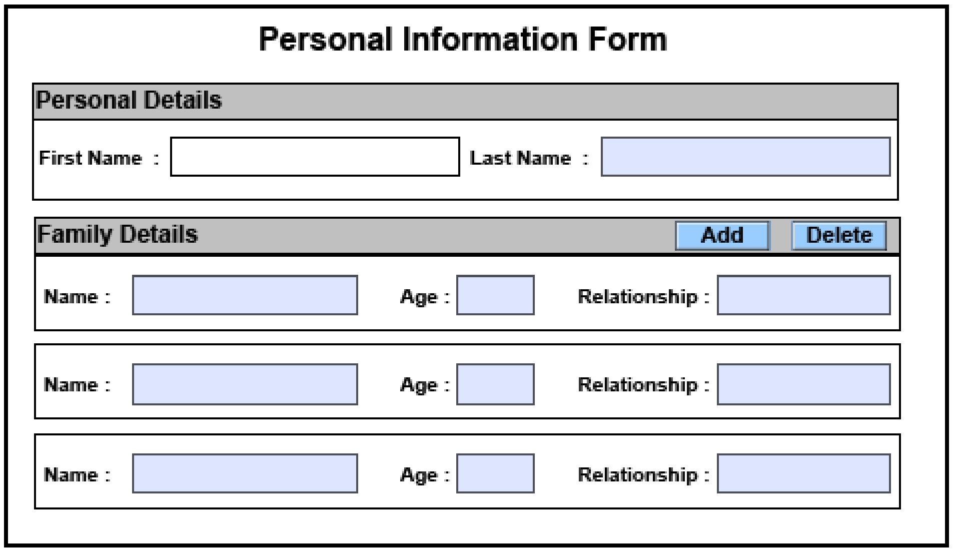


Step 3: Final step is to select the subform *Main_SubForm* from *Hierarchy* palette → select *Object* palette → click *Subform* tab → change content type to *Flowed.*



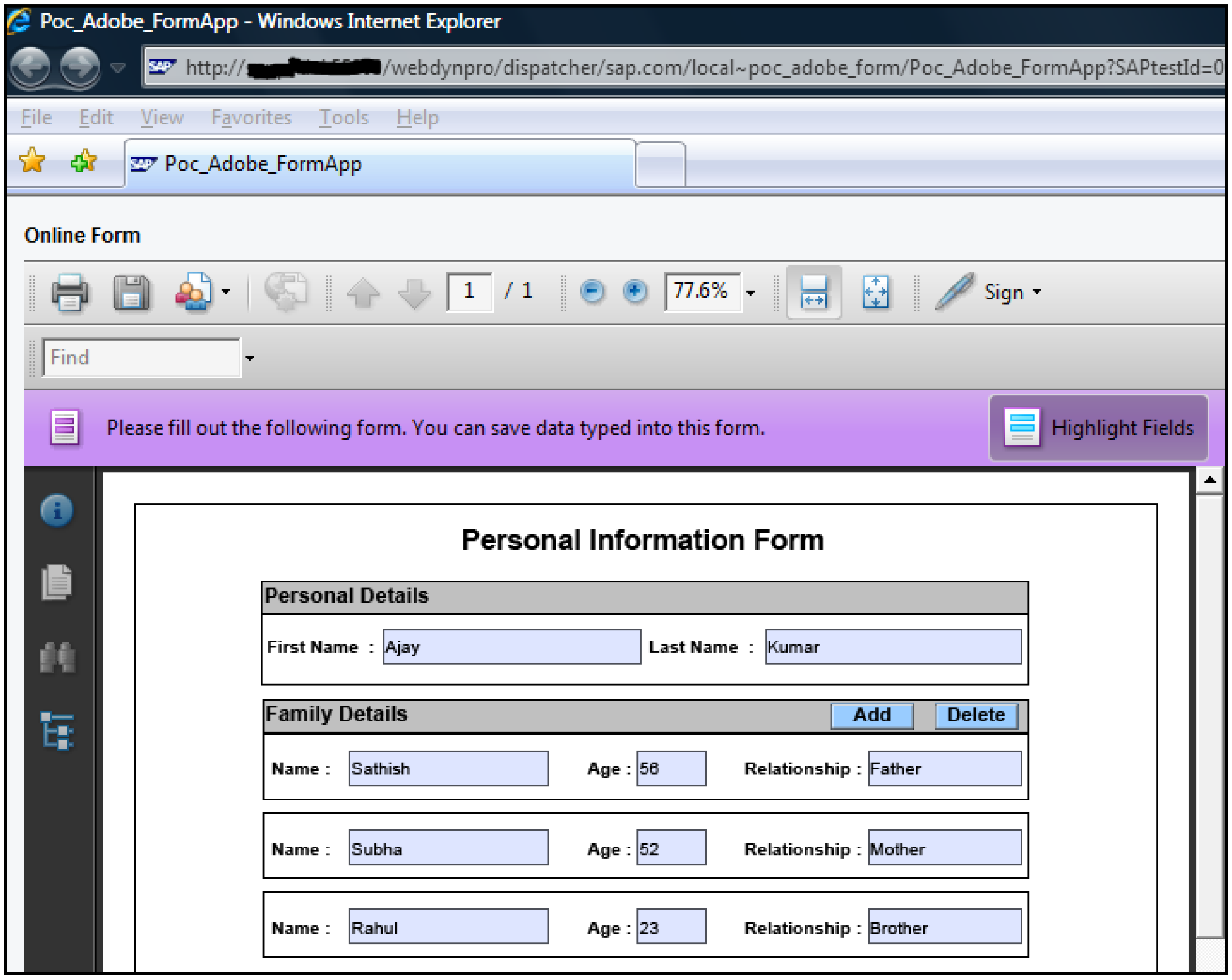
Final Output in Preview PDF

Below screen shows the output of *Preview PDF*. Clicking on 'Add' button should add a row, while clicking on 'Delete' button should delete the row from last.



Build & Deploy

Build and deploy the project from NetWeaver Developer Studio. Upon executing the application, the output in Internet Explorer browser should be similar to the screen below.


Poc_Adobe_FormApp - Windows Internet Explorer
http://[redacted]/webdynpro/dispatcher/sap.com/local~poc_adobe_form/Poc_Adobe_FormApp?SAPtestId=0
File Edit View Favorites Tools Help
Poc_Adobe_FormApp
Online Form
1 / 1
77.6%
Sign
Find
Highlight Fields

Please fill out the following form. You can save data typed into this form.

Personal Information Form

Personal Details

First Name :	Ajay	Last Name :	Kumar

Family Details Add Delete

Name :	Sathish	Age :	56	Relationship :	Father
Name :	Subha	Age :	52	Relationship :	Mother
Name :	Rahul	Age :	23	Relationship :	Brother

Note: The next article will talk about extracting data from the subform instance in both online and offline Interactive Adobe Forms.

Related Content

Adding and removing subforms at runtime

SAP Interactive Forms by Adobe

Webdynpro Java User Interface Technology

For more information, visit the Web Dynpro Java homepage.

Disclaimer and Liability Notice